

ESPECIALIDAD: TECNOLOGIA DE LA INFORMACION Y COMINICACION,

AREA SISTEMAS INFORMATICOS

MATERIA: BASE DE DATOS

MAESTRO: LIC. CARLOS GONZALEZ GONZALEZ

NOMBRE DEL ALUMNO(A): SINDY MORAN PEREZ

TRABAJO: INVESTIGACION DE TRIGGER Y PROCEDIMIENTOS ALMACENADOS

GRADO: 2 **GRUPO:** “A”



QUE ES UN TRIGGER:

Un *trigger* (o disparador) en una Base de datos , es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación. Dependiendo de la base de datos, los triggers pueden ser de inserción (INSERT), actualización (UPDATE) o borrado (DELETE). Algunas bases de datos pueden ejecutar triggers al crear, borrar o editar usuarios, tablas, bases de datos u otros objetos.

CARACTERÍSTICAS:

- No aceptan parámetros o argumentos (pero podrían almacenar los datos afectados en tablas temporales)
- No pueden ejecutar las operaciones *COMMIT* o *ROLLBACK* por que estas son parte de la sentencia SQL del disparador (únicamente a través de transacciones autónomas)
- Pueden causar errores de mutaciones en las tablas, si se han escrito de manera deficiente.

VENTAJAS:

Se crean para conservar la integridad referencial y la coherencia entre los datos entre distintas tablas.

DESVENTAJAS:

No pueden ser invocados directamente; al intentar modificar los datos de una tabla para la que se ha definido un disparador, el disparador se ejecuta automáticamente.

- no reciben y retornan parámetros.

- son apropiados para mantener la integridad de los datos, no para obtener resultados de consultas.

MODO DE EMPLEO:

Son usados para mejorar la administración de la Base de datos, sin necesidad de contar con que el usuario ejecute la sentencia de SQL.

Además, pueden generar valores de columnas, previene errores de datos, sincroniza tablas, modifica valores de una vista, etc.

Permite implementar programas basados en paradigma lógico (sistemas expertos, deducción).

3 EJEMPLOS:

```sql
CREATE TRIGGER  [nuevo vendedor]
on dbo.vendedores
after insert
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON
    -- Insert statements for trigger here
    declare @nomvendedor int
    declare @fec_nac  int
    select @nomvendedor = nomvendedor,
           @fec_nac  = fec_nac
    from inserted
    insert dbo.dddd (log_date, nomvendedor,fec_nac)
    values (getdate(),@nomvendedor, @fec_nac)
END
GO

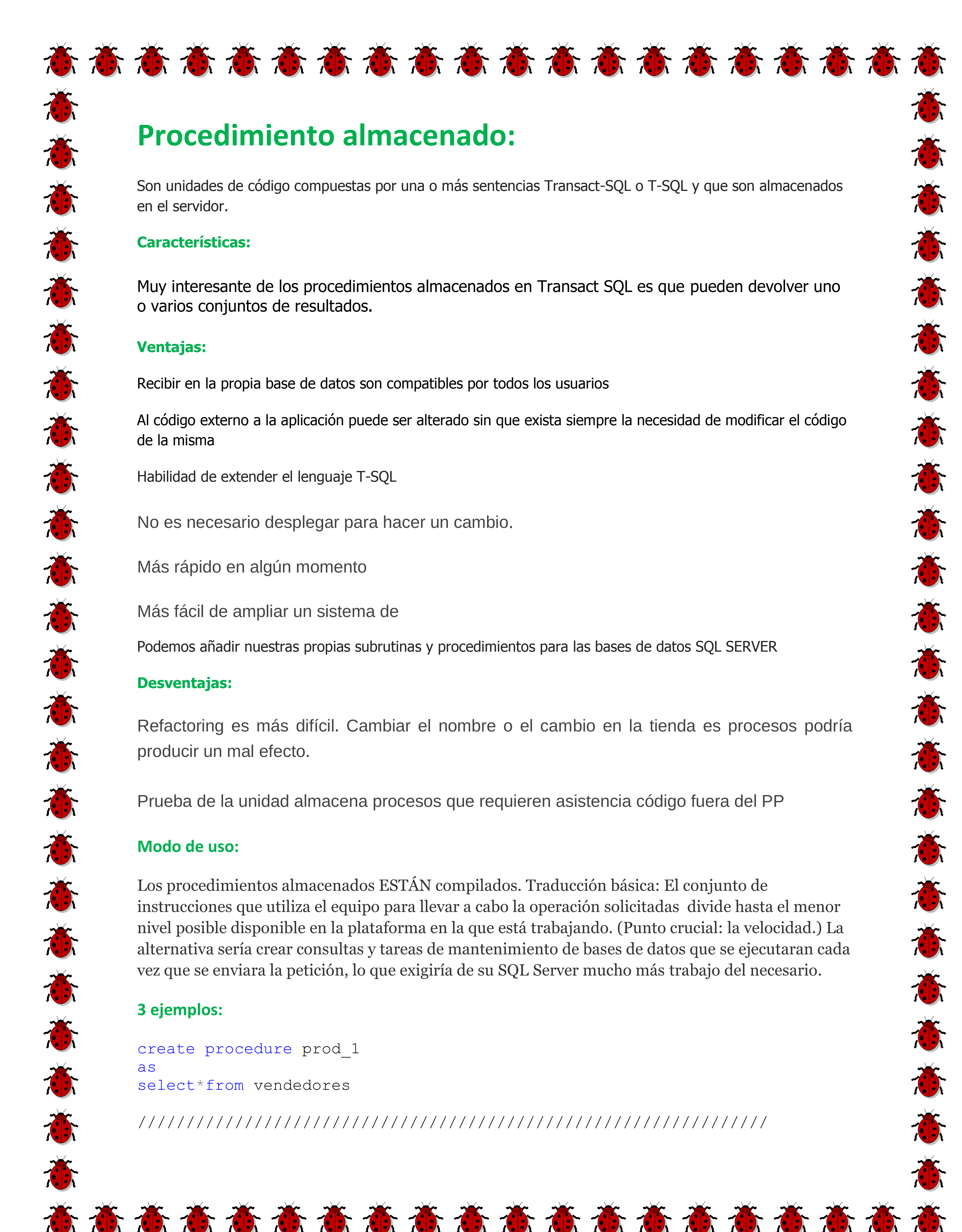
/////////////////////////////////////////////////////////////////

CREATE TRIGGER  [nuevo producto]
on dbo.productos
after insert
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;
    -- Insert statements for trigger here
    declare @nomproducto int
    declare @id_producto  int
    select @nomproducto = nomproducto,
           @id_producto = id_producto
    from inserted
    insert dbo.dddd (log_date, nomproducto, id_producto)
    values (getdate(),@nomproducto, @id_producto  )
END
GO

//////////////////////////////////////////////////////////

CREATE TRIGGER  [nuevo grupo]
on dbo.grupos
after insert
AS   BEGIN    SET NOCOUNT ON;
     declare @nomgrupo int
    declare @id_grupo  int
    select @nomgrupo = nomgrupo,
           @id_grupo  = id_grupo
    from inserted
    insert dbo.dddd (log_date, nomgrupo, id_grupo)
    values (getdate(),@nomgrupo, @id_grupo    )

END
GO
```



Procedimiento almacenado:

Son unidades de código compuestas por una o más sentencias Transact-SQL o T-SQL y que son almacenados en el servidor.

Características:

Muy interesante de los procedimientos almacenados en Transact SQL es que pueden devolver uno o varios conjuntos de resultados.

Ventajas:

Recibir en la propia base de datos son compatibles por todos los usuarios

Al código externo a la aplicación puede ser alterado sin que exista siempre la necesidad de modificar el código de la misma

Habilidad de extender el lenguaje T-SQL

No es necesario desplegar para hacer un cambio.

Más rápido en algún momento

Más fácil de ampliar un sistema de

Podemos añadir nuestras propias subrutinas y procedimientos para las bases de datos SQL SERVER

Desventajas:

Refactoring es más difícil. Cambiar el nombre o el cambio en la tienda es procesos podría producir un mal efecto.

Prueba de la unidad almacena procesos que requieren asistencia código fuera del PP

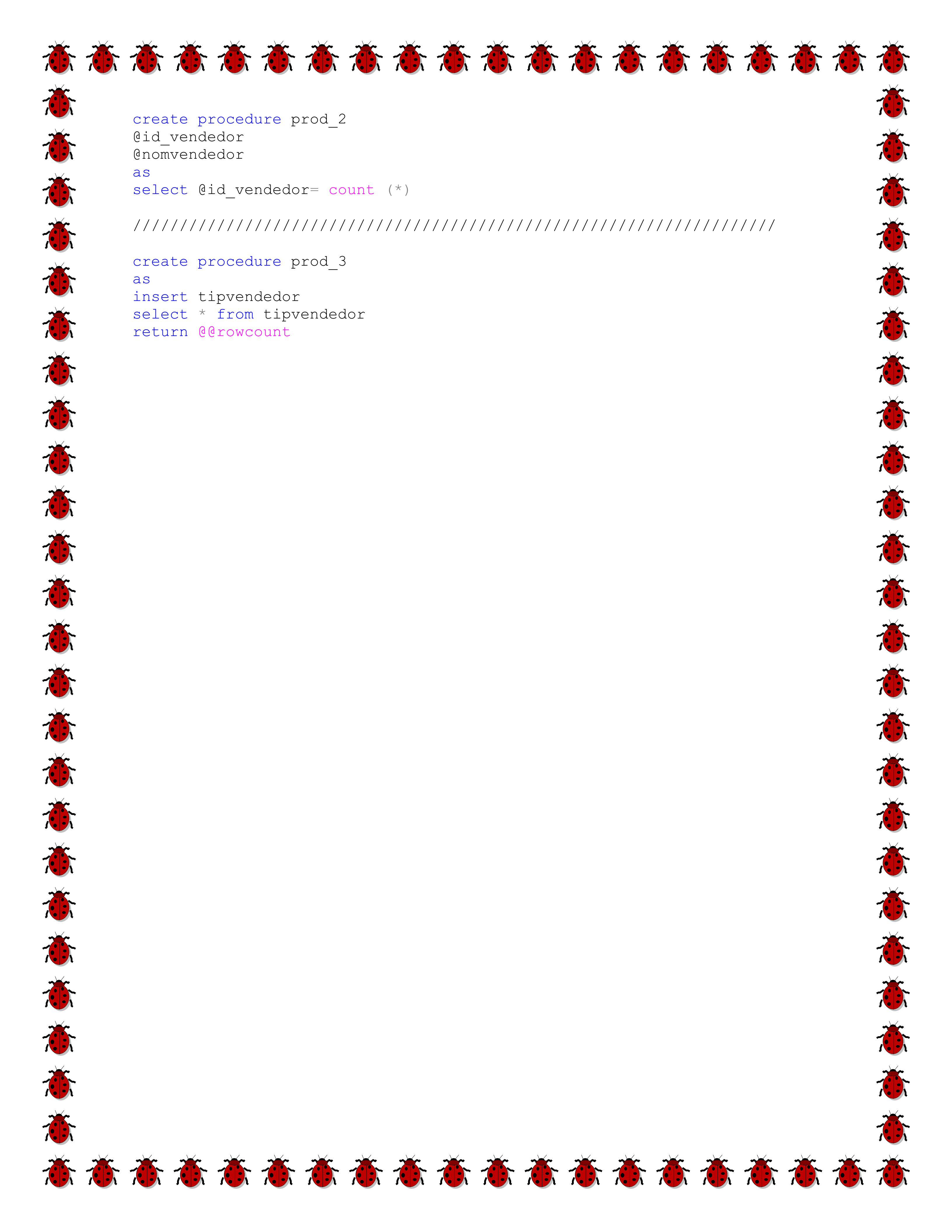
Modo de uso:

Los procedimientos almacenados ESTÁN compilados. Traducción básica: El conjunto de instrucciones que utiliza el equipo para llevar a cabo la operación solicitadas divide hasta el menor nivel posible disponible en la plataforma en la que está trabajando. (Punto crucial: la velocidad.) La alternativa sería crear consultas y tareas de mantenimiento de bases de datos que se ejecutaran cada vez que se enviara la petición, lo que exigiría de su SQL Server mucho más trabajo del necesario.

3 ejemplos:

```
create procedure prod_1
as
select*from vendedores

//////////////////////////////////////////////////////////////////////
```



```
create procedure prod_2
@id_vendedor
@nomvendedor
as
select @id_vendedor= count (*)

///////////////////////////////////////////////////////////////////////

create procedure prod_3
as
insert tipvendedor
select * from tipvendedor
return @@rowcount
```